Homework 2 Selected Solutions
APPM/MATH 4650 Fall '20 Numerical Analysis

Due date: Friday, September 11, before 5 PM, via Gradescope. **Instructor**: Prof. Becker
Theme: Convergence rates and rootfinding

solutions version 9/11/2020

Problem 1: Convergence Rates.

a) Misleading Taylor Series. To find the convergence rate of a function f, often we use Taylor series. For example, if we say $(1+x)^n = 1 + nx + O(x^2)$ as $x \to 0$, the guess of $O(x^2)$ was determined from the Taylor series $(1+x)^n = 1 + nx + n(n-1)x^2/2! + \ldots$. However, we need to be careful about the radius of convergence of the Taylor series and that the higher-order derivatives are bounded. Show that e^x is *not* $O(x)$ for $x \to \infty$.

b) Show that $x \sin \sqrt{x} = \theta(x^{3/2})$ as $x \to 0$.

Solution:

We can either show $x\sin(\sqrt{x}) = O(x^{3/2})$ and $x^{3/2} = O(x \sin(\sqrt{x}))$, or, simpler, a sufficient condition is that

$$0 < \lim_{x\to 0} \frac{x \sin(\sqrt{x})}{x^{3/2}} < \infty$$

provided the limit exists. To evaluate the limit, we can do L'Hôpital's rule

$$\lim_{x\to 0} \frac{x\sin(\sqrt{x})}{x^{3/2}} = \lim_{x\to 0} \frac{\sin(\sqrt{x})}{x^{1/2}} \overset{\text{L'H}}{=} \lim_{x\to 0} \frac{\frac{1}{2}\cos(\sqrt{x})x^{-1/2}}{\frac{1}{2}x^{-1/2}} = \lim_{x\to 0} \cos(\sqrt{x}) = 1$$

which proves the result.

Another technique that avoids L'Hôpital's rule, for showing things like $x \sin\sqrt{x} = O(x^{3/2})$, when get terms like $\sin(u)/u$ (for $u = \sqrt{x}$), then think of as $(\sin(u) - 0)/(u - 0) = \cos(\xi)$ for some $\xi \in [0, u]$ using the mean value theorem, and then use $\cos(\xi) \le 1$.

c) Show that $e^{-t} = o(\frac{1}{t^2})$ as $t \to \infty$.

Solution:

In this problem $f(t) = e^{-t}$ and $g(x) = \frac{1}{t^2}$. Then we need to look at the $\lim_{t\to\infty}$ of $\frac{f(x)}{g(x)}$. If it goes to 0, we have little o.

$$\begin{aligned} \lim_{t\to\infty} \frac{e^{-t}}{\frac{1}{t^2}} = \lim_{t\to\infty} \frac{t^2}{e^t} &= \lim_{t\to\infty} \frac{2t}{e^t} \quad \text{L'Hôpital's rule} \\ &= \lim_{t\to\infty} \frac{2}{e^t} \quad \text{L'Hôpital's rule again} \\ &= 0 \end{aligned}$$

d) Show that $\int_0^\varepsilon e^{-x^2}\,dx = O(\varepsilon)$ as $\varepsilon \to 0$. *Hint*: one nice way to do this uses the Mean Value Theorem.

Solution:

In this problem $f(\varepsilon) = \int_0^\varepsilon e^{-x^2} dx$ and $g(\varepsilon) = \varepsilon$. We need to find a positive constant M such that $\left|\frac{f(x)}{g(x)}\right| \le M$ for all x sufficiently small. Using the MVT,

$$\begin{aligned}\left|\frac{f(x)}{g(x)}\right| &= \left|\frac{\int_0^\varepsilon e^{-x^2} dx}{\varepsilon}\right| \\ &= \left|\frac{\int_0^\varepsilon e^{-x^2} dx - 0}{\varepsilon - 0}\right| \\ &= |e^{-\eta^2}| \quad \text{for some } \eta \in [0, \varepsilon] \text{ via the mean value theorem} \\ &\le 1\end{aligned}$$

Alternatively, do Taylor series of $e^u = 1 + e^\xi u$ for $|\xi| \le |u|$, so if $|u| \le 1$ then $e^u \le 1 + e^1 u$. Now substitute $u = -x^2$, so for $x \le 1$, $|e^{-x^2}| \le |1 - ex^2|$. Now, put this bound into the integral, and so

$$\left|\int_0^\varepsilon e^{-x^2}\, dx\right| \le \int_0^\varepsilon 1 - x^2\, dx = \varepsilon + e/3\varepsilon/3 = O(\epsilon)$$

e) Show that $-x/\log(x)$ is $o(x)$ but not $O(x^2)$ as $x \to 0$.

Solution:

First, show it is $o(x)$:

$$\lim_{x\to 0} \frac{-x/\log(x)}{x} = \lim_{x\to 0} \frac{-1}{\log(x)} = 0$$

Now, show it is not $O(x^2)$:

$$\lim_{x\to 0} \frac{-x/\log(x)}{x^2} = \lim_{x\to 0} \frac{-1}{x\log(x)} = \lim_{x\to 0} \frac{-1/x}{\log(x)} \overset{\text{L'H}}{=} \lim_{x\to 0} \frac{1/x^2}{1/x} = \lim_{x\to 0} 1/x$$

which diverges to $+\infty$, so it is not $O(x^2)$.

Problem 2: Write a function `bisect` which calls the bisection method, and takes as in put a function `f`, numbers specifying the interval $[a, b]$, and a tolerance `tol` which controls how far the approximate root is from the true root.

Make sure to notify the user in some way (e.g., raising an error, or returning an informative exit code) is the given function does *not* change signs on $[a, b]$. Optionally, you can also specify a maximum number of iterations.

The deliverable for this problem is your **code** for this function `bisect`. You may use any programming language, though Python and Matlab are encouraged.

Note: In Matlab, the input function might be defined in another file, like in `myfun.m`. In this case, to call your bisection function, use it like `bisect(@myfun,...)`. The `@` tells Matlab to make a function handle. Alternatively, if you defined `myfun` as an anonymous function, then you don't need the `@`, e.g., `fcn=@(x)x^2; bisect(fcn, ...)`.

Solution:

Below is a very basic code in Matlab that shows the algorithm (see end of this PDF for a nicer but slightly longer version of the code) [Python code is available at the demo Ch2_IntroToBisection.ipynb on github]:

```matlab
function [p,iter,p_hist,fp_hist] = bisect(f,a,b,tol,varargin)
oppositeSign = @(fa,fb) sign(fa)*sign(fb) < 0 ;
midpoint     = @(a,b) a + (b-a)/2;
fa = f(a);
fb = f(b);
if ~oppositeSign(fa,fb)
   error('bisect:noSignChange','f does not necessarily change signs on [a,b]');
end
d = (b-a)/2;
for iter= 1:MaxIters
   p  = midpoint(a,b);
   fp = f(p);
   if d < tol
       break
   end
   if oppositeSign(fa,fp)
       fb = fp;
       b  = p;
   else
   fa = fp;
   a  = p;
   end
   d = (b-a)/2;
end
```

Problem 3: Consider the equation $2x - 1 = \sin x$.

a) Find a closed interval $[a, b]$ on which the equation has a root r, and use the Intermediate Value Theorem to prove that r exists.

Solution:

There are many valid intervals $[a, b]$ of course. If you don't have a good idea on what to guess, then plot the function! We can use $[0, \pi/2]$ because defining $f(x) = 2x - 1 - \sin(x)$, we have $f(0) = -1$ and $f(\pi/2) = \pi - 1 - 1 > 0$, so f changes signs on $[0, \pi/2]$ and since it is also continuous, we conclude via the IVT that there exists at least one root.

b) Prove that r from (a) is the only root of the equation (on all of $\mathbb{R}$).

Solution:

Method 1: Again with $f(x) = 2x - 1 - \sin(x)$, we compute $f'(x) = 2 - \cos(x)$ and since $|\cos(x)| \le 1$, we know $f'(x) \ge 1$, and $f'(x) > 0$ means that f is a strictly monotonically increasing function. Such a function can only have one root, otherwise it violates the definition of strictly increasing.

Method 2: Suppose we have two roots r and s, so $f(r) = 0$ and $f(s) = 0$. By observation, we know 0 is not a root, so we can assume both r and s are nonzero.

Then $f(r) + f(s) = 0$, i.e., $2r - 1 - \sin(r) + 2s - 1 - \sin(s) = 0$. Rewriting we find $2(r + s) = \sin(r) + \sin(s)$. However, for $r \ne 0$, $\sin(r) < r$ and in particular $\sin(r) < 2r$, and similarly $\sin(s) < 2s$. Thus $2(r + s) > \sin(r) + \sin(s)$, which is a contradiction.

c) Use your function from Problem 2 to approximate r to eight correct decimal places. The **deliverable** is a list of the approximation at every step.

Solution:

Using the Matlab code at the end of this PDF, we get the following output:

```
>> [p,iter,p_hist,fp_hist] = bisect( @(x)2*x-1-sin(x),0,pi/2,1e-8,...
        'Print',true,'MaxIters',50);
Iter   1, p is  0.7853981633974483, f(p) is -1.36e-01, bound on error is 7.85e-01
Iter   2, p is  1.1780972450961724, f(p) is +4.32e-01, bound on error is 3.93e-01
Iter   3, p is  0.9817477042468103, f(p) is +1.32e-01, bound on error is 1.96e-01
Iter   4, p is  0.8835729338221293, f(p) is -5.86e-03, bound on error is 9.82e-02
Iter   5, p is  0.9326603190344698, f(p) is +6.21e-02, bound on error is 4.91e-02
Iter   6, p is  0.9081166264282996, f(p) is +2.79e-02, bound on error is 2.45e-02
Iter   7, p is  0.8958447801252145, f(p) is +1.10e-02, bound on error is 1.23e-02
Iter   8, p is  0.8897088569736720, f(p) is +2.53e-03, bound on error is 6.14e-03
Iter   9, p is  0.8866408953979006, f(p) is -1.67e-03, bound on error is 3.07e-03
Iter  10, p is  0.8881748761857863, f(p) is +4.28e-04, bound on error is 1.53e-03
Iter  11, p is  0.8874078857918435, f(p) is -6.22e-04, bound on error is 7.67e-04
Iter  12, p is  0.8877913809888149, f(p) is -9.70e-05, bound on error is 3.83e-04
Iter  13, p is  0.8879831285873006, f(p) is +1.66e-04, bound on error is 1.92e-04
Iter  14, p is  0.8878872547880577, f(p) is +3.43e-05, bound on error is 9.59e-05
Iter  15, p is  0.8878393178884363, f(p) is -3.13e-05, bound on error is 4.79e-05
Iter  16, p is  0.8878632863382470, f(p) is +1.47e-06, bound on error is 2.40e-05
Iter  17, p is  0.8878513021133416, f(p) is -1.49e-05, bound on error is 1.20e-05
Iter  18, p is  0.8878572942257943, f(p) is -6.73e-06, bound on error is 5.99e-06
Iter  19, p is  0.8878602902820206, f(p) is -2.63e-06, bound on error is 3.00e-06
Iter  20, p is  0.8878617883101338, f(p) is -5.79e-07, bound on error is 1.50e-06
Iter  21, p is  0.8878625373241904, f(p) is +4.46e-07, bound on error is 7.49e-07
Iter  22, p is  0.8878621628171621, f(p) is -6.67e-08, bound on error is 3.75e-07
Iter  23, p is  0.8878623500706763, f(p) is +1.90e-07, bound on error is 1.87e-07
Iter  24, p is  0.8878622564439191, f(p) is +6.14e-08, bound on error is 9.36e-08
Iter  25, p is  0.8878622096305406, f(p) is -2.66e-09, bound on error is 4.68e-08
Iter  26, p is  0.8878622330372299, f(p) is +2.94e-08, bound on error is 2.34e-08
Iter  27, p is  0.8878622213338853, f(p) is +1.34e-08, bound on error is 1.17e-08
Iter  28, p is  0.8878622154822129, f(p) is +5.35e-09, bound on error is 5.85e-09
Stopping (reached tolerance)
```

d) The function $f(x) = (x-5)^9$ has a root (with multiplicity 9) at $x = 5$ and is monotonically increasing (decreasing) for $x > 5$ ($x < 5$) and should thus be a suitable candidate for your function above. Use `a=4.82` and `b=5.2` and `tol = 1e-4` and use `bisection` with:

   i. $f(x) = (x-5)^9$.

   ii. The expanded expanded version of $(x-5)^9$, that is, $f(x) = x^9 - 45x^8 + \ldots - 1953125$. You may use `polyval` or `numpy.polyval`

The **deliverables** for this problem are (1) a graph of the error produced from both variants discussed above, and (2) a discussion of what you think is happening.

Solution:

We observe that the error for the first method decays as expected for bisection. For the second method, because this is an unstable algorithm to evaluate the polynomial (it's also just ill-conditioned), up to machine precision, we cannot accurately determine the sign of the function when it's very close to the root (recall our plot from Homework 1). So bisection method gets confused, and the error does not decrease to 0. This is one good reason to include a bound on the number of iterations in your bisection method!

Note that for the first method $f(x) = (x-5)^9$, we may loose a little bit of precision, but we always get the sign correct, since the way the computer implements z^9 it will ensure that $z > 0 \implies z^9 > 0$ and similarly for $z < 0$. For $z = x - 5$, we may lose precision doing the subtraction but we should at least get the correct sign.

Matlab code that generated our results is as follows:

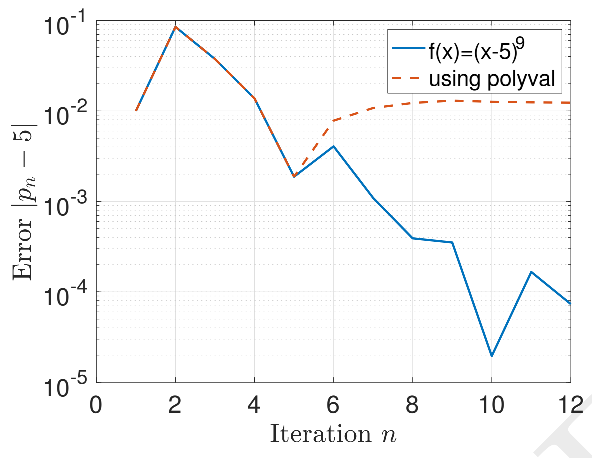

Figure 1: Plot for Problem 4. Plots like this should have a logarithmic y-axis. Showing a linear y-axis doesn't convey nearly as much information.

```
[p,iter,p_hist,fp_hist] = bisect( @(x)(x-5)^9,4.82,5.2,1e-4,'Print',true);
polyCoefficients =  poly( repmat(5,1,9) );
>> f = @(x) polyval( polyCoefficients, x );
[p,iter,p_hist2,fp_hist] = bisect( f,4.82,5.2,1e-4,'Print',true);
```

Problem 4: Tent Map. Consider the "tent function"

$$g_\mu(x) = \mu \min(x, 1-x)$$

on the interval $[0,1]$, where $0 \le \mu \le 2$. This is like an upside down and shifted absolute value function (suggestion: graph it, for both $\mu < 1$ and $\mu > 1$). We're interested in fixed points of the tent map, so solutions to $x = g_\mu(x)$. This is a favorite example used when teaching dynamical systems. *Suggestion:* Try making a "cobweb plot" of this function using, e.g., geogebra.org/m/uvsfvNDt. The software may not like functions defined with a "max" function, so you can instead use the fact that

$$\min(x, 1-x) = -\left|x - \frac{1}{2}\right| + \frac{1}{2}. \tag{1}$$

a) Show that for $\mu \in [0,2]$ that $g_\mu(x) \in [0,1]$ for all $x \in [0,1]$.

Solution:

We just need to show $0 \le \min_{x\in[0,1]} g_\mu(x)$ as well as $1 \ge \max_{x\in[0,1]} g_\mu(x)$. For the first one,

$$\min_{x\in[0,1]} g_\mu(x) = \min_{x\in[0,1]} \mu \min(x, 1-x) \le \mu \frac{1}{2} \le 1$$

because $x > \frac{1}{2} \implies 1-x < x$ so then in this region $\min(x, 1-x) = 1-x < 1-\frac{1}{2}$, and similarly for $x < \frac{1}{2}$, hence $\min(x, 1-x) \le \frac{1}{2}$ (which is also very obvious if you plot it).

Showing $1 \ge \max_{x\in[0,1]} g_\mu(x)$ is very similar.

b) Prove that for $0 \le \mu < 1$ there is a unique fixed point in $[0,1]$. Would the fixed point iteration find this fixed point? Does the contraction mapping theorem apply? *Hint:* The triangle inequality or reverse triangle inequality may be useful[1]

[1]The triangle inequality says that you cannot get from point a to point b any faster than taking a direct path. Mathematically, it is $|a-b| \le |a| + |b|$ (as well as $|a+b| \le |a| + |b|$). In one dimension, you can prove this by considering all the possible cases for the sign of a and b. In higher dimensions, the triangle inequality is always true by definition for any *norm*, since norms are required to satisfy the triangle inequality. The *reverse triangle inequality* is just the triangle inequality with a $a = a + 0$ trick, i.e., $|a| = |a+0| = |a-b+b| \le |a-b| + |b|$ so rearranging gives $\boxed{|a-b| \ge |a| - |b|}$.

Solution:

We suspect the contraction mapping theorem does hold, so let's try to use that. We need to prove g_μ is a contraction on this region. Our first tool for showing that g is a contraction is to prove $g'(x)| \le L < 1$ on our domain, but this approach fails us now because g_μ is not differentiable at $x = 1/2$. So we'll have to prove the contraction/Lipschitz condition directly.

The reverse triangle inequality gives $|a - b| \ge |a| - |b|$, and by interchanging the roles of a and b, we also have $|a - b| \ge |b| - |a|$. Since $||a| - |b||$ equals either $|a| - |b|$ or $|b| - |a|$, and in either case we have a bound, we conclude

$$||a| - |b|| \le |a - b|$$

i.e., $f(x) = |x|$ is Lipschitz continuous with derivative 1.

Now, $g_\mu(x) = -\mu f(x - \frac{1}{2}) + \frac{\mu}{2}$ according to Eq. (1). From the definition of Lipschitz continuity, we can see that if f is Lipschitz continuous with constant L, then $h(x) = \pm\mu f(x - a) + b$ is Lipschitz continuous with constant μL, since

$$\begin{aligned} |h(x) - h(y)| = |\pm\mu f(x-a) + b - (\pm\mu f(y-a) + b)| &= \mu\, |f(x-a) - f(y-a)| \\ &\le \mu L\, |(x-a) - (y-a)| \\ &= \mu L |x - y|. \end{aligned}$$

Hence we conclude g_μ is Lipschitz continuous with constant μ, and since $\mu < 1$, this means it is a contraction. Using part (a), we know it maps the domain to the domain, so we can apply the contraction mapping theorem, and conclude there is a unique fixed point. By observation, the fixed point is $p = 0$.

c) What are the fixed points if $\mu = 1$?

Solution:

When $\mu = 1$, fixed points satisfy $x = \min(x, 1 - x)$. Thus if $x \in [0, \frac{1}{2}]$, so that $x \le 1 - x$, we satisfy $x = \min(x, 1 - x)$. So all the points on the line interval $\boxed{[0, \frac{1}{2}]}$ are fixed points.

d) Prove that for $1 < \mu \le 2$ there are two fixed points in $[0, 1]$, and find these fixed points. Would the fixed point iteration find either of these fixed points? Does the contraction mapping theorem apply?

Solution:

Not only do we suspect the contraction mapping theorem might not apply on $[0, 1]$ (we can confirm this by seeing that $\mu > 1$ means we are not a contraction), but it cannot apply, since it would guarantee us a unique root, and we have two roots. The fixed point iteration is not guaranteed to find the fixed points, and as we'll see in problem (e), it often does not.

How do we prove there are two fixed points? We know g_μ is piecewise linear, e.g.,

$$g_\mu(x) = \begin{cases} g^{(1)}(x) \overset{\text{def}}{=} \mu \cdot x & x \in [0, \frac{1}{2}] \\ g^{(2)}(x) \overset{\text{def}}{=} \mu \cdot (1 - x) & x \in (\frac{1}{2}, 1] \end{cases}.$$

We're looking for the intersection of this with the line $y = x$. So $y = x$ and $y = g^{(1)}(x)$ are both lines, and using our basic Euclidean geometry, we know they can either (1) be parallel, (2) be the same line, or (3) intersect in exactly one point. Since $\mu \ne 1$ they are not the same line nor parallel, so it must be case (3) that they intersection in exactly one point, which we easily determine must be $\boxed{\text{x=0}}$.

Similarly, the intersection of $y = x$ and $y = g^{(2)}(x)$ must be at a single point. We solve $x = \mu(1 - x)$ so $\boxed{x = \frac{\mu}{1+\mu}}$ (which we can confirm is always in $[0, 1]$).

Student solutions do not need to be extremely rigorous, but they should include some explanation, and not *just* a plot.

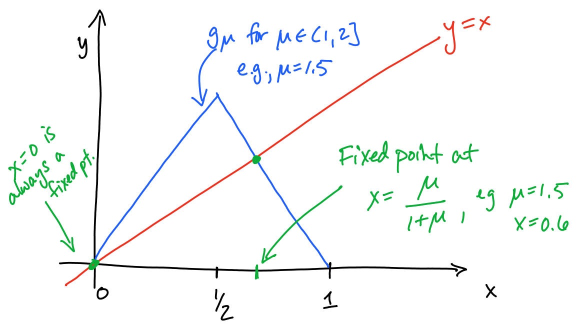
y
g_μ for $\mu \in [1,2]$
e.g., $\mu=1.5$
$y=x$
x=0 is always a fixed pt.
Fixed point at
$x=\frac{\mu}{1+\mu}$, eg $\mu=1.5$
$x=0.6$
0
½
1
x

Figure 2: Plot that helps us visualize problem 4d

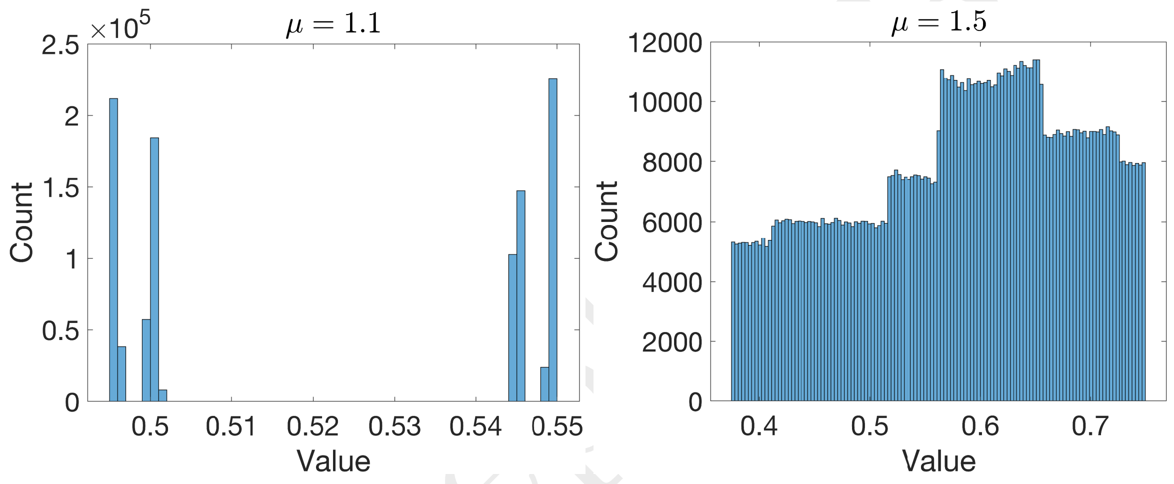

Figure 3: Plots for problem 4e

e) Run the fixed point iteration starting at $x = \pi/6$ for both $\mu = 1.1$ and $\mu = 1.5$. Do this for about 10 iterations; do you get a rough feel for what is happening? Then run 10^6 iterations and make a histogram (still for both values of μ). Include the histogram plots with your homework.

Solution:

We don't converge but rather get cyclical or aperiodic behaviors. We don't visit the entire space. For $\mu = 1.1$, we stay near a few points (note that for this μ, the fixed point is $x = 0.5238$; the iterates never get close to it, but appear to be roughly symmetrically distributed on either side, so we are alternating from being too large to too small). For $\mu = 1.5$ the fixed point is $x = 0.6$ but the iterates seem to move all over the place, distributed in a region of about $[.375, .75]$.

Here is a more complete version of the `bisect.m` code for Problem 2:

```matlab
function [p,iter,p_hist,fp_hist] = bisect(f,a,b,tol,varargin)
% [p,iter,p_hist,fp_hist] = bisect(f,a,b,tol)
%   runs the bisection method on the function f in the interval [a,b]
%   up to a tolerance tol
prs = inputParser;
addParameter(prs,'MaxIters',20); % optional fancy stuff
addParameter(prs,'Print',false); % optional fancy stuff
parse(prs,varargin{:});
MaxIters    = prs.Results.MaxIters;
```

```
printInfo   = prs.Results.Print;

oppositeSign = @(fa,fb) sign(fa)*sign(fb) < 0 ;
midpoint     = @(a,b) a + (b-a)/2;

fa = f(a);
fb = f(b);
if ~oppositeSign(fa,fb)
    error('bisect:noSignChange','f does not necessarily change signs on [a,b]');
end

d = (b-a)/2;
[p_hist,fp_hist] = deal( zeros(1,MaxIters) );
for iter= 1:MaxIters
    p  = midpoint(a,b);
    fp = f(p);
    if printInfo
        fprintf('Iter %3d, p is %19.16f, f(p) is %+5.2e, bound on error is %5.2e\n',iter,p,
            fp,d);
    end
    p_hist(iter)  = p; fp_hist(iter) = fp; % save history
    if d < tol
        p_hist = p_hist(1:iter); fp_hist = fp_hist(1:iter);
        if printInfo, fprintf('Stopping (reached tolerance)\n'); end
        break
    end
    if oppositeSign(fa,fp)
        fb = fp;
        b  = p;
    else
        fa = fp;
        a  = p;
    end
    d = (b-a)/2;
end
```